УДК 519.712.2

Е. С. Скворцов

VEGAS – НОВЫЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДЛЯ ЗАДАЧИ ВЫПОЛНИМОСТЬ

Введение

Генетические алгоритмы – это стохастический метод решения оптимизационных задач. Впервые они рассматривались в [1] и стали позже широко известными благодаря работе [2]. В генетических алгоритмах используются два основных механизма эволюции – наследование и соперничество или выживание наилучших, которое приводит к исчезновению из популяции индивидов с «плохими» свойствами. В роли индивидов, как правило, выступают элементы области определения целевой функции, а их приспособленность к окружающей среде задается значением целевой функции.

Генетические алгоритмы находят широкое применение как в практических, так и в теоретических областях компьютерных наук. Генетический подход успешно применялся для конструирования самолетов [3], маршрутизации водопроводов [4], составления расписаний [5]. В то же время на основе генетического подхода разрабатываются эффективные эвристические алгоритмы для решения NP-полных комбинаторных задач Выполнимость [6], Задача Коммивояжера [7] и др. В своем простейшем варианте генетические алгоритмы могут быть применены к любым задачам оптимизации функции нескольких переменных.

Работа генетического алгоритма решения оптимизационной задачи строится по следующей схеме. Прежде всего некоторым образом (в простейшем варианте – случайным) генерируется начальная популяция. Каждый индивид в ней – это строка, в которой закодировано потенциальное решение. Затем популяция итеративно обновляется за счет того, что в нее добавляются потомки присутствующих в ней индивидов и удаляются «погибшие» особи. Вероятность гибели индивида тем больше, чем меньше индивид приспособлен к среде, что соответствует меньшему значению максимизируемой функции.

В данной работе мы исследуем модификацию генетического алгоритма, в которой рассматривается влияние индивидов на окружающую среду. В то время как в классическом варианте популяция обитает и развивается в неизменном окружении, в предлагаемом нами алгоритме популяция и окружающая среда активно влияют друг на друга, эволюционируя вместе. Сравнение

нашего алгоритма с существующими показывает, что такое взаимное влияние существенно повышает эффективность вычисления, позволяя популяции не останавливаться на локальных максимумах целевой функции.

В качестве «полигона» для реализации этого подхода нами выбрана задача Выполнимость. Два фактора способствуют этому выбору. Во-первых, окружающая среда в генетических алгоритмах соответствует условию задачи, а значит, для реализации нашего подхода необходимо иметь возможность ее изменять, не изменяя при этом положения глобального максимума целевой функции, и для задачи Выполнимость такая возможность есть. Во-вторых, задача Выполнимость представляет большой интерес как с теоретической, так и с практической точки зрения. Это первая задача, NP-полнота которой была доказана [8]. В то же время из всех NP-полных задач именно она привлекает наибольшее внимание создателей эвристических алгоритмов. Этой задаче целиком посвящена ежегодная конференция SAT, включающая соревнование алгоритмов. Для оценки эффективности алгоритмов, решающих задачу Выполнимость, существует множество известных тестовых задач [9].

Как обычно при решении генетическими алгоритмами задачи Выполнимость, значение целевой функции мы полагаем равным количеству выполненных ограничений. Одним из лучших генетических алгоритмов, разработанных для задачи Выполнимость, является алгоритм GASAT [10].

Разработанный нами алгоритм мы назвали VEGAS (сокращенно от Valuation Enhanced Genetic Algorithm for Satisfability*). Для того чтобы оценить, насколько включение воздействия индивидов на окружающую среду влияет на эффективность вычислений, мы сравниваем эффективность VEGAS с эффективностью GASAT на ряде известных тестовых задач [9].

Проведенные тесты показывают, что VEGAS может решить больший класс задач, чем GASAT, а задачи, посильные GASAT, алгоритм VEGAS решает за меньшее время, оказываясь, таким образом, сильнее своего предшественника. В результате мы видим, что предложенная нами модификация генетической модели заслуживает внимания и исследования в случае решения и других задач.

Другим направлением нашего исследования является влияние «социальной структуры» популяции на эффективность вычислений. Как правило, социальная структура отражается на операторе выбора родителей нового индивида. Мы используем другой подход, заключающийся в том, что повышаем вероятность наилучшего индивида участвовать в размножении в роли отца и при этом нарушаем равенство вероятностей получения ребенком гена от отца и от матери. Такой подход предоставляет параметры, правильная на-

*Усиленный взвешиванием генетический алгоритм для задачи Выполнимость.

стройка которых может повысить эффективность алгоритма при решении конкретного класса задач.

Оставшаяся часть работы организована следующим образом: в разделе 1 мы приводим основные определения, в частности, даем определение задачи Выполнимость, в разделе 2 описываем алгоритм VEGAS. Раздел 3 посвящен экспериментальным результатам и сравнению алгоритмов VEGAS и GASAT. Завершает работу раздел 4, содержащий заключение и описание дальнейших возможных направлений исследования.

1. Основные определения

В этом разделе мы даем формальное определение задачи Выполнимость и описываем генетический подход к ее решению.

1.1. Выполнимость

Литералом называется булева формула, являющаяся переменной или отрицанием переменной. Дизъюнкцию литералов нам будет удобно называть *ограничением*. Говорят, что булева формула ϕ имеет *конъюнктивную нормальную форму*, если ϕ – конъюнкция ограничений.

Определение 1. *Задача Выполнимость* – это комбинаторная задача распознавания, каждая конкретная задача S которой формулируется следующим образом:

УСЛОВИЕ: Дана ϕ – булева формула в конъюнктивной нормальной форме.

ВОПРОС: Существует ли набор значений переменных, обращающий ϕ в истинное выражение?

Количество переменных в ϕ называется *размерностью* задачи и обозначается $\dim S$. Мы будем полагать, что переменные, входящие в ϕ, имеют вид x_i, где $i \in \{1, \ldots, \dim S\}$. Отметим, что размерность задачи может не совпадать с ее размером, как он определяется в теории сложности [11].

Нам будет удобно говорить о наборе значений как о булевом векторе, i-я компонента которого соответствует значению i-й переменной. Следуя терминологии генетического подхода, мы будем называть эти векторы *индивидами*.

Для краткости утверждение «c – ограничение, присутствующее в ϕ» мы будем сокращать до $c \in \phi$, а утверждение «набор значений $\vec{x}$ удовлетворяет c» – до $c(\vec{x})$.

Задача Выполнимость – это задача распознавания. Для того чтобы применить генетический подход к ее решению, необходимо свести ее к задаче

оптимизации. Этой задачей оптимизации будет задача Взвешенная Выполнимость, встречающаяся также в литературе [12] под названием MAX-SAT.

Определение 2. Задача *Взвешенная Выполнимость* – это комбинаторная задача оптимизации, каждая конкретная задача P которой формулируется следующим образом:

УСЛОВИЕ: Даны ϕ – булева формула в конъюнктивной нормальной форме и функция $v(\cdot)$, ставящая в соответствие каждому ограничению c из ϕ вес, т. е. вещественное значение $v(c) > 0$.

ВОПРОС: Для какого набора значений переменных сумма весов выполненных ограничений максимальна?

В более формальной постановке задача заключается в том, чтобы найти вектор $\vec{x}$, доставляющий максимум функции

$$\sum_{c \in \phi, c(\vec{x})} v(c)\,.$$

Отметим, что для задачи Выполнимость S требуемое сведение легко произвести, положив $v(c) \equiv 1$. Действительно, максимум целевой функции получившейся задачи S' равен количеству ограничений тогда и только тогда, когда исходная задача S имеет решение.

Определение 3. Функцию

$$V(\vec{x}) = \sum_{c \in \phi, c(\vec{x})} v(c),$$

определяемую условиями задачи Взвешенная Выполнимость, мы будем, следуя терминологии генетического подхода, называть *функцией приспособленности.*

Функция приспособленности имеет естественную биологическую аналогию. Мы полагаем, что конкретная задача Взвешенная Выполнимость соответствует некоторой среде, а булевы векторы – индивидам, в ней обитающим. Функция приспособленности показывает степень приспособленности данного индивида к данной среде.

Возможность решать задачу Взвешенная Выполнимость является преимуществом генетических алгоритмов и алгоритмов локального поиска перед полными алгоритмами, основанными на процедуре Дейвиса–Путнама [13].

Заметим, что если множество ограничений, удовлетворяемых вектором $\vec{x}$, содержится в множестве ограничений, удовлетворяемых вектором $\vec{y}$, то будет справедливо неравенство $V(\vec{x}) < V(\vec{y})$. Кроме того, в случае когда

формула ϕ выполнима, функция V достигает своего глобального максимума на множестве векторов, выполняющих ϕ вне зависимости от функции v. Это дает нам возможность в будущем изменять веса ограничений, не изменяя при этом решения соответствующей задачи Выполнимость.

Мы переходим к описанию генетического подхода к решению задачи Взвешенная Выполнимость.

1.2. Генетические алгоритмы

Генетический подход к решению задачи Взвешенная Выполнимость заключается в следующем. Пусть дана задача S и пусть V – соответствующая ей функция приспособленности. Проводя аналогию с биологией, мы полагаем, что задача S соответствует некоторой среде, и смотрим на векторы $\vec{x} \in \{0,1\}^{\dim S}$ как на генотипы обитающих в ней существ, при этом приспособленость индивида $\vec{x}$ к среде характеризуется значением $V(\vec{x})$. Таким образом, решение задачи Взвешенная Выполнимость S представляется нам как поиск наиболее приспособленного к среде S существа. Решение этой задачи осуществляется посредством моделирования эволюционного процесса.

Прежде всего мы случайным образом генерируем начальную популяцию $\mathcal{P} \subseteq \{0,1\}^{\dim S}$. Затем мы обновляем популяцию, имея целью получить в ней индивидов с более высокой приспособленностью. Для этого мы скрещиваем индивидов, присутствующих в популяции, между собой и производим отбор наиболее приспособленных, т. е. тех, для которых значение V наибольшее.

Скрещивание производится по следующей схеме. Из популяции некоторым образом (в простейшем варианте – случайным) выбираются родители будущего индивида – вектора $\vec{x}, \vec{y} \in \mathcal{P}$, а затем строится их потомок $\vec{z}$. Нам будет удобно называть вектор $\vec{x}$ отцом индивида $\vec{z}$, а $\vec{y}$ – матерью. Схема построения потомка имеет два параметра – это вероятность мутации μ и вероятность получения очередного гена от отца p. Очередной ген потомка с вероятностью μ выбирается случайно и с вероятностью $1-\mu$ наследуется от одного из родителей. Если гену выпало наследоваться от родителей, то с вероятностью p он будет унаследован от отца и с вероятностью $1-p$ от матери. Более формально,

$$z_i = C(\vec{x}, \vec{y}, p, \mu)_i = \begin{cases} 1 & \text{с вероятностью } 0,5 \cdot \mu, \\ 0 & \text{с вероятностью } 0,5 \cdot \mu, \\ x_i & \text{с вероятностью } p \cdot (1-\mu), \\ y_i & \text{с вероятностью } (1-p) \cdot (1-\mu). \end{cases}$$

(Отметим, что в нашем алгоритме каждый индивид может выступать как в роли отца, так и в роли матери.) Затем в популяцию $\mathcal{P}$ добавляется индивид

$\vec{z} = C(\vec{x}, \vec{y}, p, \mu)$ и после этого удаляется индивид $\vec{w}$, наименее приспособленный к среде, при этом $\vec{w}$ может оказаться равным $\vec{z}$. Следуя биологической терминологии, мы называем функцию C *кроссинговером*. Обычно функция кроссинговера определяется как функция трех параметров $\vec{x}, \vec{y}, \mu$. Мы добавляем параметр p, и он будет использоваться при построении социальной структуры популяции. Если положить $p \equiv 0,5$, получим «обычный» кроссинговер.

В представленном ниже формальном описании наиболее простой версии генетического алгоритма, решающего задачу Взвешенная Выполнимость S, используются следующие функции:

- $RandomIndividual(L)$ – возвращает случайный вектор длины L;
- $push(\mathcal{P}, \vec{x})$ – добавляет индивида $\vec{x}$ в популяцию $\mathcal{P}$;
- $RandomIndividual(\mathcal{P})$ – возвращает одного из индивидов популяции $\mathcal{P}$;
- $C(\vec{x}, \vec{y}, p, \mu)$ – кроссинговер, определенный выше;
- $Add(\mathcal{P}, \vec{z})$ – добавляет в популяцию $\mathcal{P}$ индивида $\vec{z}$ и удаляет из нее наихудшего индивида.

Входными параметрами являются количество индивидов в популяции N, количество итераций T, вероятность мутации μ. Через $\arg \min_{a \in A}(\max)(f(a))$ мы обозначаем $a_0 \in A$ такое, что $f(a_0) = \min(\max)\{f(a) | a \in A\}$.

Алгоритм 1. GA (N, T, μ)

do N *times* :
$push(\mathcal{P}, RandomIndividual(\dim S))$;
do T *times* :
{
  $\vec{x} = RandomIndividual(\mathcal{P})$;
  $\vec{y} = RandomIndividual(\mathcal{P})$;
  $\vec{z} = C(x, y, 0.5, \mu)$;
  $\vec{w} = \arg \min_{\vec{w} \in \mathcal{P}}(V(\vec{w}))$;
  if $(V(\vec{z}) > V(\vec{w}))$ *then* $Add(\mathcal{P}, \vec{z})$;
}
return $\arg \max_{\vec{x} \in \mathcal{P}} (V(\vec{x}))$

2. Описание алгоритма VEGAS

Предлагаемый нами алгоритм (VEGAS) является результатом модификации генетического подхода. А именно, в вычислении используется аналог такого биологического явления, как влияние индивидов на окружающую среду, а также вводится социальная иерархия популяции.

В классическом подходе, проводя аналогию с живой природой, мы рассматриваем вектор $\vec{x}$ как генетический код живого существа. Степень приспособленности индивида к окружающей среде полагается нами равной количеству ограничений, удовлетворямых вектором $\vec{x}$. При создании алгоритма VEGAS мы, как и в классическом случае, считаем вектор $\vec{x}$ генетическим кодом некоторого организма. Ограничение c мы считаем формой хищника, от которого существо с генотипом $\vec{x}$ защищено, если $\vec{x}$ удовлетворяет этому ограничению, т. е. если $c(\vec{x})$. Когда в результате размножения появляется существо с генотипом $\vec{x}$, хищники, от которых это существо не защищено, размножаются, что в терминах задачи выражается в увеличении веса ограничений, которые нарушаются. Увеличения весов невыполненных ограничений на величину δ осуществляет функция $FeedPredators$.

Алгоритм 2. FeedPredators$(\vec{x}, \delta)$
$for\ c \in \phi : \neg c(\vec{x})\ do$
$v(c) = v(c) + \delta$

Кроме того, мы исследуем также целесообразность введения социальной структуры популяции. Социальная структура популяции учитывается в алгоритме VEGAS следующим образом: индивид с наибольшим значением целевой функции считается «вожаком». С вероятностью q «вожак» выбирается отцом новой особи, и случайным образом выбирается второй родитель. В противном случае родители выбираются случайно. Таким образом, изменяя параметр q, мы влияем на привилегированность генов вожака по отношению к генам остальных особей популяции.

Авторами GASAT было показано [14], что использование улучшения индивидов при помощи широко известного в настоящее время метода TabuSearch увеличивает эффективность генетического подхода при решении задачи Выполнимость, поэтому мы сохраняем этот элемент в нашем алгоритме.

TabuSearch – это современный метод локального поиска глобального максимума целевой функции, определенной на дискретном множестве. В этом методе для избегания локальных максимумов происходит частичное запоминание истории работы. На каждом шаге выбирается наилучший из соседей текущего состояния, но, будучи однажды посещенным, состояние запрещается (становится табу) и не может быть посещено за несколько следующих шагов.

Другой вариант TabuSearch – запоминание не состояния, а направления движения. Преимуществом этого метода является существенная экономия памяти и времени на проверку запрещенности хода. В этом варианте из всех соседних состояний, находящихся по разрешенным направлениям, мы выбираем наилучшее, перемещаемся в него, добавляем в очередь запрещенных направлений то, по которому осуществлено перемещение, и удаляем из очереди самое старое. Результатом работы является лучшее посещенное состояние.

В VEGAS, как и в GASAT, алгоритм TabuSearch используется следующим образом. Выбрав родителей из популяции и получив с помощью кроссинговера нового индивида, мы передаем его в качестве начальной точки алгоритму TabuSearch. В популяцию мы добавляем результат его работы, получившийся через фиксированное число шагов.

В нашем случае состояниями являются наборы значений переменных, а направлениями – переменные. Движение по направлению – это изменение значения переменной на противоположное. Целевая функция – $V(\vec{x})$.

Для формального описания алгоритма TabuSearch введем следующее обозначение. Пусть дана задача S, а V – соответствующая ей функция приспособленности. Положим по определению

$$\frac{\partial V}{\partial i}(\vec{x}) = V(x_1, \ldots, x_{i-1}, \neg x_i, x_{i+1}, \ldots, x_{dimS}) - V(\vec{x}).$$

Таким образом, $\frac{\partial V}{\partial i}(\vec{x})$ – изменение числа выполненных ограничений при изменении i-й координаты $\vec{x}$.

Также нам понадобится объект TabuQueue – очередь, элементами которой являются номера позиций. Для нее определены три операции. Операция $push(i)$ добавляет элемент в конец очереди, операция pop удаляет элемент из начала очереди, а операция $length$ возвращает количество элементов в ней.

Параметрами процедуры TabuSearch являются следующие величины:

- $\vec{x}$ – в начале работы – вектор с которого начинается поиск, в конце – результат поиска;
- λ – отношение максимальной длины списка запрещенных состояний к размерности задачи;
- n – количество шагов поиска.

Алгоритм 3. TabuSearch$(\vec{x}, \lambda, n)$
$\vec{b} = \vec{x}$;
$TabuQueue = \varnothing$;
do n *times* :

$\{$
$\quad i = \arg \max\limits_{i \notin TabuQueue} \left(\frac{\partial V}{\partial i}(\vec{x})\right)$;
$\quad x_i = \neg x_i$;
$\quad push(TabuQueue, i)$;
$\quad if\ (length(TabuQueue) / \dim(S) > \lambda)\ then\ pop(TabuQueue)$;
$\quad if\ (V(\vec{x}) > V(\vec{b}))\ then\ \vec{b} = \vec{x}$;
$\}$
$\vec{x} = \vec{b}$

Одним из параметров работы алгоритма VEGAS является величина δ, на которую увеличивается вес не выполненных новым индивидом ограничений. Применяя алгоритм VEGAS, мы можем решать либо задачу Выполнимость, либо задачу Взвешенная Выполнимость. Если решается задача Взвешенная Выполнимость, то мы в начале вычисления запомним веса v в v_0, полагая $v_0(c) = v(c)$ для каждого c, если же решается задача Выполнимость, то мы инициализируем $v_0 \equiv 1$. В ходе вычисления мы запоминаем индивида $\vec{b}$, максимизирующего значение $V_0(\vec{b}) = \sum\limits_{c \in \phi, c(\vec{b})} v_0(c)$, как наилучшего из встреченных. Кроме того, мы можем положить $\delta = \frac{1}{T}$ (где T – количество итераций эволюционного процесса), и в этом случае в ходе всего вычисления будет выполняться неравенство $V(\vec{x}) > V(\vec{y})$ для всех $\vec{x}, \vec{y}$, для которых $V_0(\vec{x}) > V_0(\vec{y})$.

Ниже дано формальное описание алгоритма VEGAS. Помимо δ его параметрами являются:

- N – размер популяции;
- T – число итераций;
- λ_0 – отношение длины списка табу к длине генетического кода индивидов при инициализации популяции;
- n_0 – количество шагов алгоритма TabuSearch при инициализации популяции;
- λ – отношение длины списка табу к длине генетического кода индивидов в ходе эволюции;
- n – количество шагов алгоритма TabuSearch в ходе эволюции;
- p – вероятность того, что очередной ген ребенка будет взят у отца;
- q – вероятность того, что в роли отца очередного ребенка будет выступать «вожак»;

- μ – вероятность мутации.

Алгоритм 4. $\mathbf{VEGAS}(\mathbf{N}, \mathbf{T}, \lambda_\mathbf{0}, \mathbf{n_0}, \lambda, \mathbf{n}, \delta, \mathbf{p}, \mathbf{q}, \mu)$;

$for\ c \in \phi\ do\ v(c) = v_0(c) = 1;$
$\vec{b} = RandomIndividual(\dim S);$
$do\ N\ times:$
{
  $\vec{x} = RandomIndividual(\dim S);$
  $TabuSearch(\vec{x}, \lambda_0, n_0);$
  $push(\mathcal{P}, \vec{x});$
}
$do\ T\ times:$
{
  $$\vec{x} = \begin{cases} \arg\ \max\limits_{\vec{x}\in\mathcal{P}}(V(\vec{x})), & \textit{с вероятностью } q, \\ RandomIndividual(\mathcal{P}), & \textit{с вероятностью } 1-q; \end{cases}$$
  $\vec{y} = RandomIndividual(\mathcal{P});$
  $\vec{z} = C(x, y, p, \mu);$
  $TabuSearch(\vec{z}, \lambda, n);$
  $if\ (V_0(\vec{z}) > V_0(\vec{b}))\ then\ \vec{b} = \vec{z};$
  $FeedPredators(\vec{z}, \delta);$
  $\vec{w} = \arg\ \min_{\vec{w}\in\mathcal{P}}(V(\vec{w}));$
  $if\ (V(\vec{z}) > V(\vec{w}))\ then\ Add(\mathcal{P}, \vec{z});$
}
$return\ \vec{b}$

В экспериментальной части мы проводим сравнение эффективности алгоритмов VEGAS и GASAT. А также исследуем влияние социальной структуры популяции на эффективность вычисления.

3. Результаты экспериментов

3.1. Сравнение эффективности VEGAS и GASAT

Мы проводим сравнение эффективности VEGAS и GASAT на тех же самых примерах, что использовали авторы GASAT для оценки своего алгоритма. Это задачи, участвовавшие в разные годы в соревнованиях алгоритмов, решающих задачу Выполнимость [9]:

- aim* – случайным образом сгенерированные задачи, имеющие единственное решение [15];

- par* – задачи распознавания зависимости [16];
- color* – задача раскраски шахматной доски [17];
- dp* – задача проверки модели – сгенерирована на основе известного примера «обедающие философы» [14];
- mat* – задачи об умножении матриц посредством минимального числа произведений [18];
- g* – задачи раскраски графа.

Задачи серий dp* и mat* – неразрешимы, и для этих формул нами решается задача оптимизации Взвешенная Выполнимость с весами всех ограничений, равными единице.

Таблица 1. Сравнение эффективности алгоритмов VEGAS и GASAT

Задачи			VEGAS		GASAT		GASAT(авт)	
Файл	$\dim S$	$\lvert\phi\rvert$	Успех	Время	Успех	Время	Успех	Время
aim-100-1_6-yes1-1.cnf	100	160	100%	0,43	0	(1)	2%	2
aim-200-1_6-yes1-1.cnf	200	320	85%	2,48	0	(1)	–	–
par8-1-c.cnf	64	254	100%	0,06	10%	0,17	100%	0,028
par8-1.cnf	350	1149	100%	1,04	10%	41,87	17%	8,01
par8-2.cnf	350	1157	85%	1,45	0	(1)	25%	11,12
par32-5-c.cnf	1339	5350	0%	(4)	0	(13)	0%	(5)
par32-5.cnf	3176	10325	0%	(8)	0	(129)	0%	(16)
color-15-4.cnf	900	45675	100%	14,91	100%	15,22	100%	479,248
color-22-5.cnf	2420	272129	40%	255,45	0	(3)	0%	(5)
dp11u10.shuffled.cnf	9197	25271	0%	(2)	0	(81)	0%	(36)
mat25.shuffled.cnf	588	1968	0%	(3)	0	(8)	0%	(39)
mat26.shuffled.cnf	744	2464	0%	(2)	0	(8)	0%	(56)
g125.18.cnf	2250	70163	100%	5,92	100%	30,19	98%	92,0
g250.29.cnf	7250	454622	5%	313,08	0	(2)	0%	(2)

В колонке $|\phi|$ указано количество ограничений в задаче, а в колонке *время* – среднее время вычисления (с), закончившегося решением задачи, если задача была решена хотя бы один раз. Если решение не было найдено, то в скобках указывается число невыполненных ограничений для наилучшего найденного индивида. В таблице приведена результативность версии GASAT,

выложенной на сайте (колонка GASAT), и показаны результаты, полученные самими авторами (колонка GASAT(авт)).

В табл. 1 мы сравниваем эффективность алгоритмов VEGAS и GASAT. Настройки алгоритма GASAT были выбраны в соответствии с рекомендацией его авторов. Мы проводим сравнение с отключенной социальной структурой, поэтому устанавливаем значения параметров $p = 0,5$ и $q = 0$. Остальные параметры были нами изначально положены равными параметрам GASAT, однако ввиду специфики VEGAS, т. е. влияния индивидов на окружающую среду, оказывается целесообразным увеличить количество кроссинговеров с 500 до 5000. Для того чтобы этим не давать преимущества VEGAS, мы уменьшаем число шагов Tabu Search с 10000 до 300. Таким образом, количество просматриваемых решений алгоритмом VEGAS будет равняться $T_{\text{VEGAS}} \times n_{\text{VEGAS}} = 5000 \times 300 = 1.5 \times 10^6$ и будет меньше количества решений, просматриваемых GASAT: $T_{\text{GASAT}} \times n_{\text{GASAT}} = 500 \times 10000 = 5 \times 10^6$. Кроме того, наши тесты показали, что эффективность VEGAS увеличится, если уменьшить размер популяции со 100 индивидов до 10. На работе GASAT изменение этого параметра ощутимым образом не сказывается.

Настройки для VEGAS и GASAT, используемые в наших тестах, приведены в табл. 2.

Таблица 2. Используемые нами настройки алгоритмов GASAT и VEGAS

Параметр	GASAT	VEGAS
N	100	10
T	500	5000
λ_0	0.1	0.1
n_0	10000	10000
λ	0.1	0.1
n	10000	300
p	–	0.5
q	–	0
δ	–	0.1
μ	0.02	0.02

Тестирование проводилось на персональном компьютере Pentium IV 3GHz. Результаты тестов GASAT, полученные нами, отличаются от тех, что публикуют авторы в [14] и [10]. Это, по всей видимости, объясняется различиями версий алгоритма, опубликованной на сайте, и той, что использовали авторы при написании работы. Мы приводим оба набора результатов, чтобы читатель имел возможность убедиться в качественном сходстве работы обеих версий.

Обратимся к табл. 1. В частности, можно заметить следующее. Из 11 задач, обладающих удовлетворяющим вектором, GASAT решил 4, в то время как VEGAS нашел решения для 9. Алгоритм VEGAS решает задачи из серии aim и задачи малой размерности из серии par за небольшое время с высокой вероятностью, в то время как GASAT решает их с малой вероятностью или находит лишь их приближенные решения. В таблице мы приводим только по две задачи из этих серий. Для задач $mat25$, $mat26$ и $dp11-10$, условия которых невыполнимы, алгоритму VEGAS удается найти более точные приближения, чем GASAT.

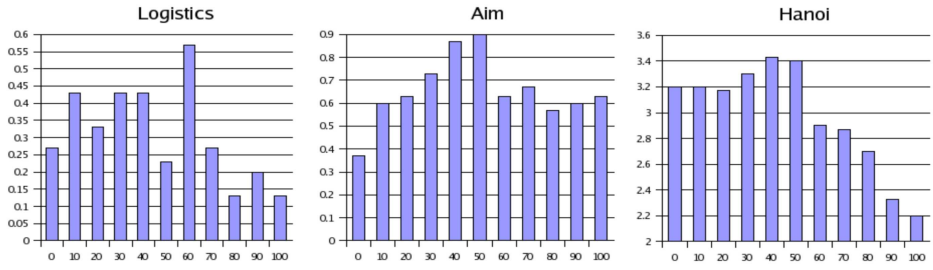
3.2. Исследование влияния социальной структуры популяции на эффективность вычисления

В [10] исследовано влияние длины списка табу на эффективность вычисления. В данной работе исследовано влияние вероятности получения новым индивидом гена от отца на эффективность вычисления при условии, что в качестве отца выбирается наилучший индивид. Для этого были проведены вычисления при фиксированных параметрах $N = 10$, $n_0 = 10000$, $\lambda_0 = 0.1$, $n = 100$, $\lambda = 0.1$, $T = 5000$, $q = 1$, и параметре p, изменяющемся от 0 до 1 с шагом 0.1. При этом время вычисления было ограничено 10 с, а величиной, характеризующей эффективность вычислений, мы считали количество ограничений, не удовлетворенных наилучшим найденным вектором, которое мы усреднили по 20 вычислениям для каждого значения p.

Таблица 3. Зависимость среднего числа неудовлетворенных ограничений от вероятности получения гена отца при 100 % вероятности выбора в качестве отца вожака

значение	$\lvert\{c \lvert c \in \phi\}\rvert - V(\vec{x})$		
p	logistics.a	aim200	hanoi
0	0,27	0,37	3,2
0,10	0,43	0,6	3,2
0,20	0,33	0,63	3,17
0,30	0,43	0,73	3,3
0,40	0,43	0,87	3,43
0,50	0,23	0,9	3,4
0,60	0,57	0,63	2,9
0,70	0,27	0,67	2,87
0,80	0,13	0,57	2,7
0,90	0,2	0,6	2,33
1,00	0,13	0,63	2,2

Таблица 4. Графики зависимости среднего числа неудовлетворенных ограничений от вероятности получения гена отца при 100 % вероятности выбора в качестве отца вожака



Здесь мы приводим результаты этих экспериментов для сильно отличающихся друг от друга по размеру и сложности задач $aim200$, $logistics.a$ (транспортная задача) и $hanoi$ (задача о ханойских башнях), на которых влияние различных настроек на эффективность наиболее ярко. Результаты тестов для них представлены в табл. 3, 4.

В результатах тестов мы наблюдаем неоднозначность влияния параметра p на эффективность вычисления. Для задачи $hanoi$ оптимальным является значение $p = 1$, что соответствует эволюционированию единственного индивида. В случае задачи aim оптимальным является $p = 0$, т. е. эволюция идет с максимальной скоростью в случае отсутствия социальной структуры в популяции. А в задаче $logistics.a$ оптимальное значение $p = 0,8$ соответствует эволюции с нетривиальной социальной структурой.

Таким образом, мы видим, что в то время как рассматриваемая нами социальная структура популяции может повысить эффективность вычисления, единого оптимального значения параметров в случае необходимости решения широкого класса задач может не существовать.

Разработка алгоритма выбора оптимальной социальной структуры, а также оптимальной настройки других параметров по ходу вычисления является темой наших будущих исследований.

4. Заключение и дальнейшие направления исследования

Мы представили VEGAS – генетический алгоритм для решения задачи Выполнимость, в котором популяция и среда взаимно влияют друг на друга и эволюционируют совместно. Также представлены результаты его тестирования и сравнения с GASAT – известным генетическим алгоритмом, решающим задачу Выполнимость. Сравнение показало, что VEGAS получает решение для задач быстрее, чем это делает GASAT, а также может решить

некоторые задачи, для которых GASAT за приемлемое время может найти лишь приближенное решение.

Кроме того, в рассматриваемом алгоритме мы исследовали эффективность моделирования социальной структуры популяции. Вычислительные эксперименты показали, что оптимальные параметры социальной структуры отличаются от задачи к задаче, это ставит цель для будущих исследований – создать алгоритм выбора по ходу вычисления оптимальной социальной структуры популяции и других параметров вычисления. На базе VEGAS также планируется создание алгоритма, решающего более общую комбинаторную задачу – задачу CSP.

1. Holland J. H. Adaptation in natural and artificial systems. Cambridge, Mass.: MIT Press, 1975.

2. Goldberg D. E. Genetic algorithms in search, oprimization, and machine learning. Reading, Mass.: Addison-Wesley, 1989.

3. Bramelette M. F., Bouchard E. E. Handbook of genetic algorithms. N. Y.: Van Nostrand Reinhold, 1991.

4. Simpson A. R., Dandy G. C., Murphy L. J. Genetic algorithms compared to other techniques for pipe optimization // J. Water Resources Planning and Management. 1994. Vol. 120, № 4. P 423–443.

5. Hiroaki U., Ouchi D., Takahashi K. et al. A co-evolving timeslot/room assignment genetic algorithm technique for university timetabling // Practice and Theory of Automated Timetabling III. [Lecture Notes in Computer Science; Vol. 2079]. B.; Heidelberg: Springer, 2001. P. 48.

6. Bertoni A., Carpentieri M., Campadelli P. et al. A genetic model: Analysis and application to MAX-SAT // Evolutionary Computation. 2000. Vol. 8, № 3. P. 291–310.

7. Han H., Xiaowei Y., Zhifeng H. et al. Hybrid chromosome genetic algorithm for generalized traveling salesman problems // Advances in Natural Computation [Lecture Notes in Computer Science; Vol. 3612]. B.; Heidelberg: Springer, 2005. P. 137–140.

8. Cook S. A. The complexity of theorem-proving procedures // The 3rd IEEE Symposium on Foundations of Computer Science, FOCS'71. IEEE Computer Society, 1971. P. 151–158.

9. Hoos H., Stützle T. Satlib: An online resource for research on SAT // SAT2000: Highlights of Satisfiability Research in the year 2000. Kluwer Academic, 2000. P. 283–292.

10. Lardeux F., Saubion F., Hao J. K. A hybrid genetic algorithm for the satisfiability problem // The 1st Int. Workshop on Heuristics. 2002. P. 69–77.

11. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

12. Du D. Z., Pardalos P. M. Handbook of Combinatorial Optimization. Boston: Kluwer Academic Publishers, 1998.

13. Moskewicz M., Madigan C., Zhao Y. et al. Chaff: Engineering an efficient SAT solver // The 38th Design Automation Conference. 2001. P. 530–535.

14. Hao J. K., Lardeux F., Saubion F. Evolutionary computing for the satisfiability problem // Applications of Evolutionary Computing. [Lecture Notes in Computer Science; Vol. 2611]. B.; Heidelberg: Springer, 2003. P. 258–267.

15. Asahiro Y., Iwama K., Miyano E. Random generation of test instanses with controlled attributes // DIMACS Series on Discrete Mathematics and Theoretical Computer Science. 1996. Vol. 26. P. 377–394.

16. Thornton C. Parity: the problem that won't go away // Advances in Artificial Intelligence. [Lecture Notes in Computer Science; Vol. 1081]. B.; Heidelberg: Springer, 1996. P. 362–374.

17. Lardeux F., Saubion F., Hao J. K. A chessboard coloring problem for SAT solver // Technical report. LERIA, Université d'Angers, 2003.

18. Li C. M., Jurkowiak B., Purdom P. Integrating symmetry breaking into DLL procedure // The 5th Int. Symposium on Theory and Applications of Satisfiability Testing. 2002. P. 149–155.

Статья поступила 21.05.2006, окончательный вариант 04.04.2008